УДК 004.4(043.2)

МОДИФІКАЦІЯ АЛГОРИТМУ ПРИСКОРЕНОГО МОДУЛЯРНОГО МНОЖЕННЯ МОНТГОМЕРІ ДЛЯ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ З ВІДКРИТИМ КЛЮЧЕМ

О. П. Марковський[1, а], В. М. Солонець[1, б]

[1] *Національний технічний університет України «Київський політехнічний інститут»*

Анотація

В роботі запропоновано та досліджено модифікацію алгоритму Монтгомері модулярного множення, що використовує передобчислення при фіксованому модулі. Показано, що продуктивність розробленого алгоритму приблизно вдвоє вища в порівнянні з алгоритмом Монтгомері при прийнятних об'ємах потребуємої пам'яті.

Ключові слова: комп'ютерна арифметика, модулярне множення, модулярне експоненціювання, передобчислення, протоколи захисту інформації в мережах.

Вступ

Однією з домінуючих тенденцій сучасного етапу розвитку інформаційних технологій є інформаційна інтеграція на основі комп'ютерних мереж. Важливою передумовою ефективної взаємодії обчислювальних термінальних пристроїв в комп'ютерних мережах є забезпечення високої продуктивності при реалізації ними існуючих мережевих протоколів. Найбільш критичними з точки зору обчислювальної реалізації протоколів мережевого обміну є операції модулярної арифметики, що виконуються над числами великої розрядності, яка значно перевищує розрядність процесорів [1]. На теперішній час, для досягнення прийнятного для більшості застосувань рівня захищеності, необхідна довжина чисел становить 1024-2048 розрядів з перспективою її зростання в найближчі роки до 4096.

Швидкий розвиток технологій хмарних "обчислень", що надають будь-якому користувачу значні за обсягом обчислювальні ресурси віддалених потужних багатопроцесорних систем призводить до зростання можливостей порушення захисту. Ця обставина вимагає адекватного підвищення рівня захищеності криптографічних протоколів, в основі яких операції модулярного експоненціювання, в першу чергу, за рахунок збільшення розрядності чисел.

Зростання розрядності чисел, з якими оперують існуючі протоколи захисту інформації в мережах призводить до сповільнення їх реалізації на комп'ютерах загального призначення. З особливою гостротою ця проблема стоїть для малорозрядних вбудованих мікропроцесорів та мікроконтролерів.

Таким чином, проблема прискорення реалізації операції модулярного експоненціювання для систем захисту інформації є актуальною.

1. Постановка задачі

Вважаючи на важливість ефективної реалізації протоколів захисту інформації, що базуються на операціях модулярної арифметики, до теперішнього часу запропоновано ряд алгоритмів, найбільш відомими з яких є класичний та Монтгомері [2].

Для модулярного множення $R = A \cdot B \bmod M$, вважається, що $2^{n-1} \le M < 2^n, A < M, B < M$. При обробці n-розрядних на k-розрядному процесорі кожне з чисел A, B і M складаються з $s = n/k$ слів:

$$A = \sum_{j=0}^{s-1} a_j \cdot 2^{j \cdot k}, B = \sum_{j=0}^{s-1} b_j \cdot 2^{j \cdot k}, M = \sum_{j=0}^{s-1} m_j \cdot 2^{j \cdot k},$$

де a_j, b_j, m_j – k-розрядні слова, $j \in \{0, ..., s-1\}$.

Операція модулярного експоненціювання, тобто обчислення $A^E \bmod M$, складається з виконання k циклів, на кожному з яких виконується піднесення до модулярного квадрату результату попереднього циклу та модулярного множення на A, якщо поточний двійковий розряд коду експоненти E дорівнює одиниці. Таким чином, модулярне експоненціювання – базова операція більшості мережевих протоколів захисту інформації складається, в середньому з $1.5 \cdot k$ операцій модулярного множення. Вважаючи на послідовний характер процедури модулярного експоненціювання, воно не може обчислюватися паралельно і, відповідно, основним резервом зменшення часу його обчислення є прискорення виконання його базової операції – модулярного множення.

Модулярне множення складається з обчислення добутку $A \cdot B$ та модулярної редукції – віднаходження залишку $A \cdot B$ по модулю M. Всі алгоритми відрізняються способом виконання модулярної редукції.

В класичному алгоритмі модулярна редукції виконується з використанням операції цілочисельного ділення $2 \cdot k$-розрядного діленого на k–розрядний подільник з одержанням частки та залишку. Операція

[а] markovskyy@i.ua
[б] victor.solonets@gmail.com

цілочисельного ділення потребує для своєї реалізації значних за обсягом обчислювальних ресурсів.

Щоб уникнути операції ділення багаторозрядних чисел при реалізації модулярної редукції запропоновано ряд алгоритмів, найбільш відомими з яких є алгоритми Баретта та Монгомері.

Найбільш ефективно модулярна редукція реалізується в алгоритмі Монтгомері: шляхом переходу від редукції за модулем M до редукції за числом, що є степенем 2.

Застосування модулярного множення за Монтгомері вимагає додаткових операцій, що виконуються до і після власне множення. На практиці модулярне множення з використанням алгоритму Монтгомері доцільне лише я якості базової операції модулярного експоненціювання.

Вихідними даними для модулярного множення по Монтгомері являються: модуль M, співмножники A і B, а також попередньо обчислена модулярна інверсія $\acute{M} = -M^{-1} \bmod 2^k$. В процесі виконання алгоритму ітераційно формується n-розрядний код результату $R = A \cdot B \cdot U \bmod M$, де U – модулярна інверсія числа 2^n за модулем M, тобто $U = (2n)^{-1} \bmod M$. Сам алгоритм Монтгомері можна представити з використанням нотацій C++ в наступному вигляді:

1. $R = 0$.
2. $for(j = 0; j \le s-1; j++)$
   {
   2.1. $p_j = (r_0 + a_j \cdot b_0) \cdot \acute{M} \bmod 2^k$
   2.2. $R = (R + a_j \cdot B + p_j \cdot M)/2^k$
   }
3. $if(R \ge M) R = R - M$

Щоб отримати правильний результат $X \cdot Y \bmod M$ потрібно виконати додаткові обчислення над R: помножити одержаний результат на $2^n \bmod M$.

Проведений аналіз [3] алгоритму Монтгомері показав, що його обчислювальна складність модулярного множення, що виконується за цим алгоритмом, без урахування додаткових перетворень близька до теоретичного мінімуму. Проте при постійному модулі його обчислювальна складність може бути зменшена. При практичному застосуванні систем захисту інформації з відкритим ключем таких як RSA та DSS, останній міняється достатньо рідко. Відповідно, рідко міняється і модуль. Це дозволяє прискорити реалізацію модулярних операцій за рахунок використання передобчислень, що залежать тільки від модуля [4].

Ціллю досліджень є зменшення обчислювальної складності модифікація алгоритму Монтгомері для випадку постійного модуля.

2. Модифікований алгоритм Монтгомері

Аналіз операцій, що складають алгоритм Монтгомері показує, що в основному циклі має місце дублювання операції множення $a_j \cdot b_0$: воно виконується в п.2.1. та повторюється п.2.2 в процесі множення k–розрядного коду a_j на n-розрядний код $B = \{b_{s-1}, ..., b_1, b_0\}$. Це дублювання може бути виключено наступним чином. Розділимо код суми $r_0 + a_j \cdot b_0$ на дві k-розрядні складові: $g_j = (r_0 + a_j \cdot b_0) \bmod 2^k$ и $d_j = (r_0 + a_j \cdot b_0 - g_j)/2^k$ так, що $p_j = d_j \cdot 2^k + g_j$. З урахуванням цього, $p_j = (d_j \cdot 2^k + g_j) \cdot \acute{M} \bmod 2^k = g_j \cdot \acute{M} \bmod 2^k$. Відповідно: $p_j \cdot M = g_j \cdot \acute{M} \cdot M - t \cdot M \cdot 2^k$, де t – ціле число. В силу того, що $M^{-1} = -M \bmod 2^k$, очевидно, що $\acute{M} \cdot M = c \cdot 2^k - 1$, де c - ціле. Тоді $p_j \cdot M = g_j \cdot c \cdot 2^k - g_j - t \cdot M \cdot 2^k = h_j \cdot 2^k - g_j$, причому $h_j = g_j \cdot c - t \cdot M$. З урахуванням цих перетворень, обчислення п.2.2. базового алгоритму Монтгомері можуть бути представлені таким чином:

$$R = (R + a_j \cdot B + p_j \cdot M)/2^k = (\sum_{i=1}^{s-1} r_i \cdot 2^i + r_0 +$$

$$+ a_j \cdot \sum_{i=1}^{s-1} b_i \cdot 2^i + a_j \cdot b_0 + h_j \cdot 2^k - g_j)/2^k =$$

$$= (\sum_{i=1}^{s-1} r_i \cdot 2^{j-1} + a_j \cdot \sum_{i=1}^{s-1} b_i \cdot 2^{i-1} + h_j +$$

$$+ (r_0 + a_j \cdot b_0 - g_j)/2^k = \acute{R} + a_j \cdot \acute{B} + h_j +$$

$$+ (d_j \cdot 2^k + g_j - g_j)/2^k = \acute{R} + a_j \cdot \acute{B} + h_j + d_j$$

$\acute{R}$ являє собою $(n-k)$-розрядний код, який може бути отриманий зсувом R на k розрядів праворуч: $\acute{R} = \{r_{s-1}, ..., r_2, r_1\} = \lfloor R/2^k \rfloor$, $(n-k)$-розрядний код $\acute{B}$ отримується зсувом множеного B праворуч на k розрядів: $\acute{B} = \{b_{s-1}, ..., b_2, b_1\} == \lfloor R/2^k \rfloor$. Складова $a_j \cdot \acute{B}$ являє собою добуток k–розрядного слова множника a_j - на код:

$$\acute{B} = \{b_{s-1}, ..., b_2, b_1\} : a_j \cdot (B/2^k) = \sum_{i=1}^{s-1} a_j \cdot b_i \cdot 2^{i-1}$$

При фіксованому модулі M значення h на кожному циклі алгоритму залежить тільки від $g_j = (r_0 + a_j \cdot b_0)\ mod\ 2^k$, тобто h може бути представлене у вигляді:

$$h = \lfloor ((g \cdot \acute{M} \bmod 2^k) \cdot M)/2^k \rfloor$$

Таким чином, значення h можуть бути попередньо обчислені для всіх 2^k можливих g і збережені у вигляді таблиці $T(g)$.

З урахуванням введених перетворень пропонується модифікований алгоритм модулярного множення на основі редукції Монтгомері: Передобчислення: $\acute{M} = -M^{-1} \bmod 2^k, T(g) = \lfloor ((g \cdot \acute{M} \bmod 2^k) \cdot M)/2^k \rfloor$ для всіх $g \in \{0, 1, ..., 2^k - 1\}$.

1. $R = 0$.
2. $for(j = 0; j \le s-1; j++)$
   {
   2.1. $v_j = r_0 + a_j \cdot b_0; g_j = v_j \bmod 2^k; d_j = \lfloor v_j/2^k \rfloor$;
   2.2. $R = \lfloor R/2^k \rfloor + a_j \cdot \lfloor B/2^k \rfloor + T(g_j) + d_j$;
   }
3. $if(R \ge M) R = R - M$

Об'єм V_1 пам'яті таблиці передобчислень становить $2^k \cdot (s-1)$ k-розрядних слів і може бути зменшений шляхом фрагментації таблиці.

Обчислювальна складність модулярного множення за запропонованим алгоритмом без урахування передобчислень визначається s^2 операціями проце-

сорного множення і $s \cdot (2 \cdot s + 3)$ операціями додавання:

$$T_1 = s^2 \cdot t_m + s \cdot (2 \cdot s + 3) \cdot t_a = \\ = s \cdot t_a \cdot (w \cdot s + 2 \cdot s + 3), \text{ де}$$

t_m - час виконання операції множення

t_a - час виконання операції додавання

Теоретично, обчислювальна складність модулярного множення $A \cdot B \bmod M$ не може бути меншою за складність обчислення $A \cdot B$, яка визначається s^2 операціями множення і $2 \cdot s^2$ операціями додавання. Таким чином, обчислювальна складність запропонованого алгоритму модулярного множення, якщо враховувати тільки операції множення, співпадає з теоретичним мінімумом.

В порівнянні з базовим алгоритмом Монтгомери обчислювальна складність запропонованого алгоритму є меншою в γ раз:

$$\gamma = \frac{T_M}{T_1} = \frac{2 \cdot s^2 \cdot (w+1) + w \cdot s}{s^2 \cdot (w+2) + 3 \cdot s} \approx 2$$

Така ефективність запропонованого алгоритму практично може бути досягнута лише для процесорів розрядності 8 або 16, коли об'єм пам'яті таблиць передобчислень не великий.

Для процесорів більшої розрядності (32 або 64) об'єм пам'яті таблиць виходить за рамки ефективної реалізації і практичне використання запропонованого методу можливе лише за умови розділення k–розрядного вхідного коду таблиць передобчислень на q фрагментів. Такий підхід може бути на практиці реалізовано у двох варіантах, що дозволяє значно зменшити об'єм пам'яті для зберігання таблиць передобчислень, але потребує додаткових обчислень.

Аналіз результатів проведених експериментальних досліджень показав, що ефективність запропонованого підходу при використанні фрагментації таблиць передобчислень суттєвим чином залежить від співвідношення часу реалізації на конкретному процесорі операцій множення та додавання. Наприклад, при реалізації модулярного множення на 32-розрядному процесорі, на якому команда множення виконується вп'ятеро повільніше ніж операція додавання, запропонований спосіб дозволяє реально прискорити модулярне множення в 1.2 рази в порівнянні з базовим алгоритмом Монтгомері, при тому, що об'єм пам'яті таблиць передобчислень складає 128 кбайтів.

Висновки

В результаті проведених досліджень розроблено модифікацію алгоритму Монтгомері для виконання модулярного множення при постійному модулі - базової операції широко кола криптографічних алгоритмів з відкритим ключем. Показано, що при постійному модулі, що є частиною відкритого ключа, запропонована модифікація алгоритму модулярного множення дозволяє приблизно вдвічі зменшити час виконання цієї операції за рахунок використання таблиць передобчислень.

Перелік використаних джерел

1. Анісімов А. В. Алгоритмічна теорія великих чисел. -К.: Академперіодика. — 2001. — 153 с.
2. Montgomery P. L. Modular multiplication without trial division. // Mathematics of Computation, Vol. 44. — 1985. — p. 519-521.
3. Koc C. K., Acar T., Kaliski B. S. Analyzing and comparing Montgomery Multiplication Algorithms.// IEEE Micro. V.16, No. 3. — 1996. — p. 26-33.
4. Boyko V., Pienado M., Venkatesan R. Speeding up log and factorization based schemes via precomputations // Advances in cryptography –Proceeding of EUROCRYPT'98, LNCS-658, Springer-Verlag. — 1998. — p. 221-253.